Remarks

Status of application

Claims 1-15, 17-40 and 42-70 were examined and stand rejected in view of prior art. The claims have been amended to further clarify Applicant's invention. In view of the amendments made and the following remarks, reexamination and reconsideration are respectfully requested.

Prior art rejections

Applicant's claims 1-15, 17-40 and 42-70 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Srivastava, et. al “Optimizing Multi-Join Queries in Parallel Relational Databases”, in Proceedings of the Second International Conference of Parallel and Distributed Information Systems, Los Alamitos, California, December 1993 (hereinafter "Srivastava") in view of U.S. Patent 7,047,530 of Lu (hereinafter "Lu"). The Examiner's rejection of Applicant's claim 1 is representative:

> As per claim 1, Srivastava teaches
> In a database system, a method for parallel optimization of a query, the method comprising: (see abstract)
> generating a plurality of parallel plans for obtaining data requested by the query, the parallel plans including parallel operators for executing portions of the query in parallel; (query plan space, section 4.1, page 87, first paragraph)
> adjusting parallel operators of each parallel plan if necessary; (for each tree and all subtrees, section 4.1, page 87, first paragraph; note that 'if' denotes an optionally recited limitation and optionally recited limitations are not guaranteed to take place and are therefore not required to be taught, see MPEP § 2106 Section II(C))
> creating a schedule for each parallel plan indicating a sequence for execution of operators of each parallel plan, wherein the schedule is created based upon dependencies among operators of each parallel plan (ordered tree where shape represents intra-operator parallelism, section 2, page 85, first paragraph) and resources available for executing the query; (query optimization considers resources available, section 6, page 91, first paragraph)
> determining execution cost of each parallel plan based on its schedule. (combining operator costs, section 3.2, page 87)
> and returning a result of a particular parallel plan having lowest execution cost for obtaining data requested by the query. (query plan representation for expressing intra and inter-operator parallelism, processor and memory assignment, and execution time estimate, section 6; page 91, first paragraph)
> Srivastava does not explicitly indicate "based on maximum number of threads available for executing the query, wherein said maximum number of threads is

> user configurable".
> However, Lu discloses "based on maximum number of threads available for executing the query, wherein said maximum number of threads is user configurable" (maximum number of threads, column 6, lines 23-27; maximum threads configured appropriately, column 8, lines 5-9; column 15, lines 25-32).
> It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine Srivastava and Lu because using the steps of "based on maximum number of threads available for executing the query, wherein said maximum number of threads is user configurable" would have given those skilled in the art the tools to improve the invention by allow parallel compilation using makefiles. This gives the user the advantage of being able to control aspects of the compilations via the makefile.

At the outset, it is important to understand that Applicant does not purport to have invented the broad notion of executing query plans in parallel. A number of different approaches exist for generating plans, including parallel plans, and choosing the best plan for executing a query. Therefore, the general notion of executing query plans in parallel is understood in the industry and documented in prior art references such as Srivastava. However, the focus of Applicant's invention is a specific improvement for optimizing query plans that may execute in parallel, by creating a schedule for each plan based on available resources and then comparing the alternative plans so as to choose the best plan. Applicant's way of generating a plurality of different parallel plans, adjusting the plans to account for available resources, creating a schedule for execution of each of these plans (including introducing additional parallelism when doing so will improve performance), and determining the best parallel plan amongst those plans differ substantially from those prior art approaches. The particular features of these specific improvements which distinguish Applicant’s invention from the prior art of record are set forth as claim limitations in Applicant's claims and are discussed in detail below.

By way of background, selecting the best plan for execution of a query in a parallel processing environment is a complex task as the underlying SQL operations may be performed in a multitude of different ways. For example, a parallel processing system could scan a given table serially or in parallel. As the scan is being performed in parallel, the result set from the scan could be merged into one stream that is funneled into a distinct operation or the data streams could be re-split into a larger number of sub-streams so that the distinct operation can be done with a greater degree of parallelism. Similar choices also need to be made for various other operations, including whether to perform a grouping operation serially or in parallel, and so forth and so on. Applicant's invention addresses these complexities by providing for creation of a schedule for each parallel plan indicating the sequence for execution of operators of each parallel plan and associated costs in a manner that is not taught or suggested by the prior art.

Applicant's solution generates a plurality of parallel plans, structured as trees of operators, for executing a given query. The operators included in such trees refer to constructs that reflect SQL operations, such as join operations, predicate evaluation mechanisms, scans (e.g., table or index), grouping operations, union operations, distinct operations -- that is, operations defined by SQL. In this regard, Srivastava discusses representing a parallel query plan which S as "capacitated labeled ordered binary tree" (Srivastava, Section 2, page 85). However, as noted below Srivastava indicates that only queries consisting of join operations are considered. Additionally, Srivastava states that developing a parallel cost model is a major task that is beyond the scope of the paper (Srivastava, Section 3.1, page 86). Applicant's solution specifically attempts to address some of these complex problems that were recognized (but not solved) by Srivastava by examining operators in an operator tree and the available resources to determine whether performing particular operations in parallel will yield improved performance in executing the query. The particular focus of Applicant's invention is finding out the best possible parallel plan given the resources that are available for executing the query.

Applicant's solution includes features for taking into account the resources which are available and adjusting the parallel plans that are generated to account for the resources (e.g., memory and processors) available for executing the query. Applicant's approach considers two general kinds of resources during schedule generation that are referred to as preemptable resources (PS) and non-preemptable resources (NPS). More particularly, Applicant's invention uses the concept of multidimensional PS and NPS resources in developing a cost model with sufficient flexibility for parallel query execution. Query operators are represented as pairs of vectors with one dimension per PS and NPS resource, respectively. Preemptable resources include disks, CPU, network interfaces, and the like (Applicant's specification, paragraph [118]). Non-preemptable resources include memory buffers, whose time-sharing among operators introduces prohibitively high overheads. Thus, the inclusion of NPS requirements gives rise to trade-offs. In some instances increasing the degree of parallelism of an operator reduces the NPS requirement, while decreasing it allows the parallelism to be kept fairly coarse-grained so that the communication cost does not overwhelm the execution (Applicant's specification, paragraph [120]).

Applicant's solution provides for specifying and maintaining an NPS capacity requirement throughout the query execution process (Applicant's specification, paragraph [120]). More particularly, during the scheduling process, a determination is made as to whether the maximum NPS resource usage of a pipeline exceeds the maximum NPS resource that is available (Applicant's specification, paragraph [155]; Fig. 12A at 1206). If the NPS resource usage of the pipeline is greater than the NPS resource available, operators are added to materialize the pipeline into a plurality of pipelines, each of which are within the NPS capacity constraints (Applicant's specification, paragraph [156], Fig. 12A at 1207). In other words, Applicant's claimed invention provides for by introducing additional parallelism into a plan based on applicable resource constraints. Applicant’s claims have been amended to bring these distinctive features to the forefront. For example, Applicant’s amended claim 1 includes the following claim limitations:

> In a database system, a method for parallel optimization of a query, the method comprising:
> generating a plurality of parallel plans for obtaining data requested by the query, the parallel plans including parallel operators for executing portions of the query in parallel;
> adjusting parallel operators of each parallel plan if based on maximum number of threads available for executing the query, wherein said maximum number of threads is user configurable;
> creating a schedule for each parallel plan indicating a sequence for execution of operators of each parallel plan, wherein the schedule is created based upon dependencies among operators of each parallel plan and resources available for executing the query and includes separating a resource intensive operator into a plurality of operators;
> determining execution cost of each parallel plan based on its schedule; and
> returning a result of a particular parallel plan having lowest execution cost for obtaining data requested by the query.
>
> (Applicant’s amended claim 1, emphasis added)

As illustrated above, Applicant's scheduling process includes examining dependencies among operators of the plans and resource constraints and costs. For each operation Applicant's system may decide to perform the operation serially or in parallel. Given that, there is a wide range of possibilities available in how serial and parallel operations (including different levels of parallelism within a given operation) are combined. There is also another dimension of choices available in how a given operation is performed. For example, a scan operation may be performed using a table scan or an index scan (as available). For a distinct operation, the system may perform a hash-based distinct operation or sort-based distinct operation. Similarly, for grouping, the system could do a hash-based grouping or sort-based grouping, or if the data is already sorted within the system, it could perform the grouping on the already-sorted data. As should be readily apparent, combining these dimensions of possibilities in all possible ways yields n different plans, where n may be a very large number.

For all of those possible plans, Applicant's system provides for computing a cost to determine which of the possible plans is the best plan. In determining the cost (elapsed time) for each possible plan, Applicant's system figures out how all the various operators are to be scheduled. As previously described, resource constraints and costs are considered in the process of creating and evaluating a schedule. The schedule indicates exactly when an operator would start and when would it stop, based on available resources (Applicant's specification, paragraph [136]; Figs. 7A-B at 705-712). Based on that information, that is the activation schedule of various operators, Applicant's system can determine the elapsed time for each particular plan (operator tree), and thus can determine which plan is in fact the best one. If the elapsed time of the current operator tree being considered is less than the minimum cost (i.e., elapsed time) of any previously evaluated operator tree, the current operator tree is saved as the best plan (best operator tree) and the elapsed time of the current plan is made the new minimum (Applicant's specification, paragraph [137]; Fig. 7B at 713-715).

The Examiner references Srivastava for the corresponding teachings of generating a schedule for execution of each operator tree based on available resources. However, Srivastava acknowledges that because of the “wide variety of parallel architectures available”, developing analytical cost expressions for them is a “major task” beyond the scope of the Srivastava paper (Srivastava, Section 3.1, page 86). Thus, Srivastava makes various assumptions about the queries that are received, including that the queries consist only of join operations and that there is enough main memory to hold the smaller relation of all joins (Srivastava, Section 3.1, page 86). Thus, while Applicant's scheduling methodology includes evaluation of the actual resources that are available, Srivastava simply assumes that sufficient resources are available and admits that it cannot handle queries when this assumption does not hold true (Srivastava, Section 3.1, page 86). Additionally, Applicant's review of Srivastava finds that it makes no mention of separating a resource intensive plan operator into a plurality of operators as provided in Applicant's specification and claims. As noted above, Srivastava simply assumes memory is sufficient and does not evaluate whether or not sufficient memory is available for performing a given operation. Thus, it is clear that Srivastava's teachings are not comparable to Applicant's claim limitations of evaluating available resources (e.g., memory) and separating a resource intensive operator into a plurality of operators when doing so will provide improved query processing performance.

Another difference between Applicant's claimed invention and the prior art references is that Applicant's claimed invention provides for adjusting an operator tree based on available threads (worker processes). This illustrates another manner in which Applicant's invention adjusts the degree of parallelism of a plan based on available resources. Here, each of the operator trees (plans) is adjusted for available worker processes (i.e., threads) (Applicant's specification, paragraph [134]). This adjustment ensures that an operator tree (plan) does not exceed the maximum number of configured worker processes. It should be noted here that Applicant has amended its independent claims to remove the conditional "if necessary" language so as to address the Examiner's argument that this adjustment of the operator tree based on available threads and, therefore, is not required to be taught by the prior art references.

The Examiner acknowledges that Srivastava does not including this teaching of adjusting plans based on maximum threads available for executing the query and therefore adds Lu for these teachings. Lu, however, describes a different parallel processing problem in a very different context; namely, the context of compiling source code of an application that includes thousands of separate source code files (Lu, col. 2, lines 1-2). The particular problem addressed by Lu's invention is that prior make file utilities operate serially even on multi-processor machines (Lu, col. 2, lines 5-13). In this context, Lu provides a command server that is adaptable to provide for an acceptable range of parallel compilation which limits the maximum number of threads that can be spawned for compiling selected targets (Lu, col. 6, lines 18-26). Significantly, Lu does not mention adjusting a query execution plan to ensure that an operator tree (plan) does not exceed the maximum number of configured worker processes. In fact, the Lu reference makes no mention of a query or an operator tree (plan) for executing a query given that it relates to parallel compilation of any application composed of multiple source code files which is a fundamentally different problem than that addressed by Applicant's claimed invention.

All told, the prior art references do not include teachings of adjusting operator trees (plans) based on threads and other resources available for executing a given query. Additionally, the prior art references do not describe anything comparable to Applicant's scheduling process which generates a schedule for activation of operators of a particular plan based on dependencies among operators and available resources and which separates a given operator that is resource intensive into multiple operators when necessary as a result of resource limitations. Therefore, as Srivastava and Lu, even when combined, do not include all the limitations of Applicant's claims it is respectfully submitted that Applicant's claims distinguish over the prior art references and overcome any rejection under Section 103.

Any dependent claims not explicitly discussed are believed to be allowable by virtue of dependency from Applicant's independent claims, as discussed in detail above.

Conclusion

In view of the foregoing remarks and the amendment to the claims, it is believed that all claims are now in condition for allowance. Hence, it is respectfully requested that the application be passed to issue at an early date.

If for any reason the Examiner feels that a telephone conference would in any way expedite prosecution of the subject application, the Examiner is invited to telephone the undersigned at 925 465 0361.

Respectfully submitted,

Date: May 7, 2008 /G. Mack Riddle/

G. Mack Riddle; Reg. No. 55,572
Attorney of Record

925 465 0361
925 465 8143 FAX